ROS2 – 11 เขียน Node ภาษา C++ แบบ OOP



เขียน Node ภาษา C++ แบบ OOP  
  
OOP ก็คือ “ธรรมชาติของวัตถุ” หมายความว่า OOP จะมองสิ่งแต่ละสิ่งถือเป็น “วัตถุชิ้นหนึ่ง” (Object) มันจะมีสีแดงหรือสีเขียว ยาวหรือสั้น มันก็คือวัตถุชิ้นหนึ่งเหมือนกัน และเราสามารถกำหนดประเภทหรือคลาสให้กับวัตถุเหล่านั้นได้ นอกจากนี้ เมื่อ OOP มองทุกสิ่งถือเป็นวัตถุชิ้นหนึ่งแล้ว ยังสามารถคิดต่อไปอีกว่า “วัตถุแต่ละอย่างนั้น ต่างก็มีลักษณะและวิธีการใช้งานเป็นของตัวเอง” หมายความว่า วัตถุแต่ละชนิดหรือแต่ละชิ้นต่างก็มีรูปร่าง ลักษณะ และการใช้งาน (การกระทำ) ที่แตกต่างกันออกไป เราจะเรียกคุณลักษณะของวัตถุว่า แอตทริบิวต์ (Attribute) และจะเรียกวิธีการใช้งานวัตถุว่า เมธทอด (Method)

**1 : เขียน Node ภาษา C++ เบื้องต้น**  
เขียน Node ภาษา C++ เบื้องต้น ตามขั้นตอนลิงค์ด้านล่าง

* [เขียน Node ภาษา C++ เบื้องต้น](http://www.lungmaker.com/ros2-10-%e0%b9%80%e0%b8%82%e0%b8%b5%e0%b8%a2%e0%b8%99-node-%e0%b8%a0%e0%b8%b2%e0%b8%a9%e0%b8%b2-c-%e0%b9%80%e0%b8%9a%e0%b8%b7%e0%b9%89%e0%b8%ad%e0%b8%87%e0%b8%95%e0%b9%89%e0%b8%99/)

**2 : เขียน Node ภาษา C++ แบบ OOP**  
ใช้ Visual Studio Code เปิดไฟล์ my\_first\_node.cpp เขียนโค้ด + Save ตามโค้ดด้านล่าง

#include "rclcpp/rclcpp.hpp"

class MyNode : public rclcpp::Node

{

public:

MyNode() : Node("cpp\_test"), counter\_(0)

{

RCLCPP\_INFO(this->get\_logger(), "Hello Cpp Node");

timer\_ = this->create\_wall\_timer(std::chrono::milliseconds(1),

std::bind(&MyNode::timerCallback, this));

}

private:

void timerCallback()

{

counter\_++;

RCLCPP\_INFO(this->get\_logger(), "Hello %d", counter\_);

}

rclcpp::TimerBase::SharedPtr timer\_;

int counter\_;

};

int main(int argc, char \*\*argv)

{

rclcpp::init(argc, argv);

auto node = std::make\_shared<MyNode>();

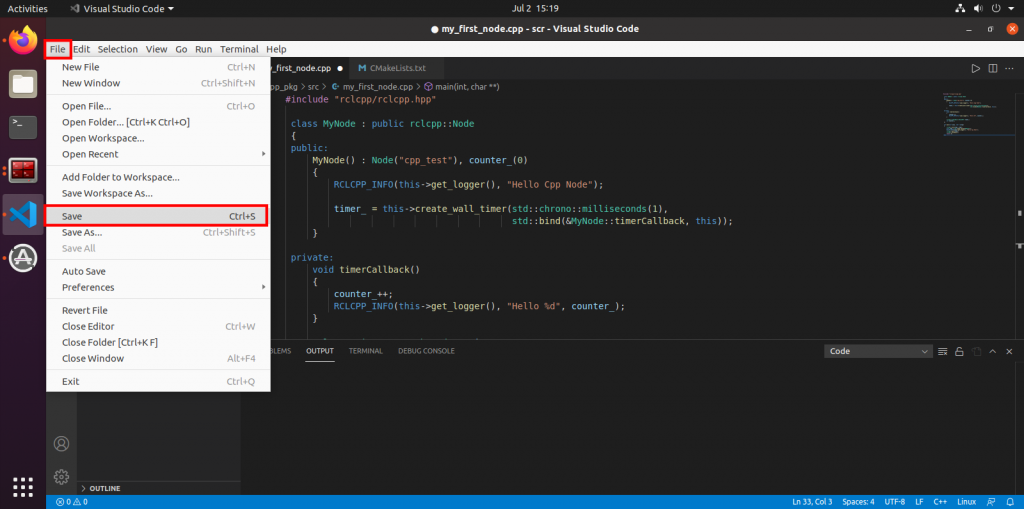
RCLCPP\_INFO(node->get\_logger(), "Hello Cpp Node");

rclcpp::spin(node);

rclcpp::shutdown();

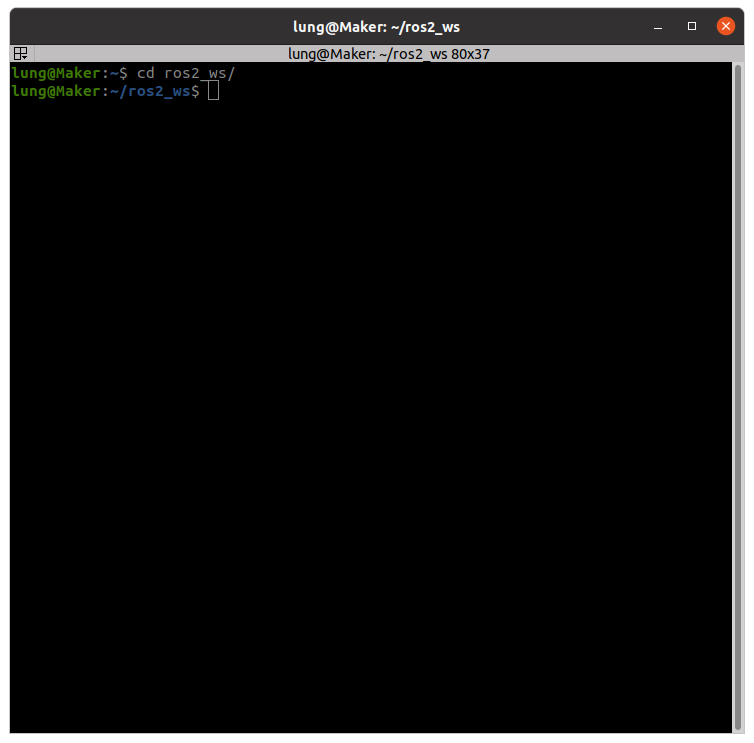
return 0;

}



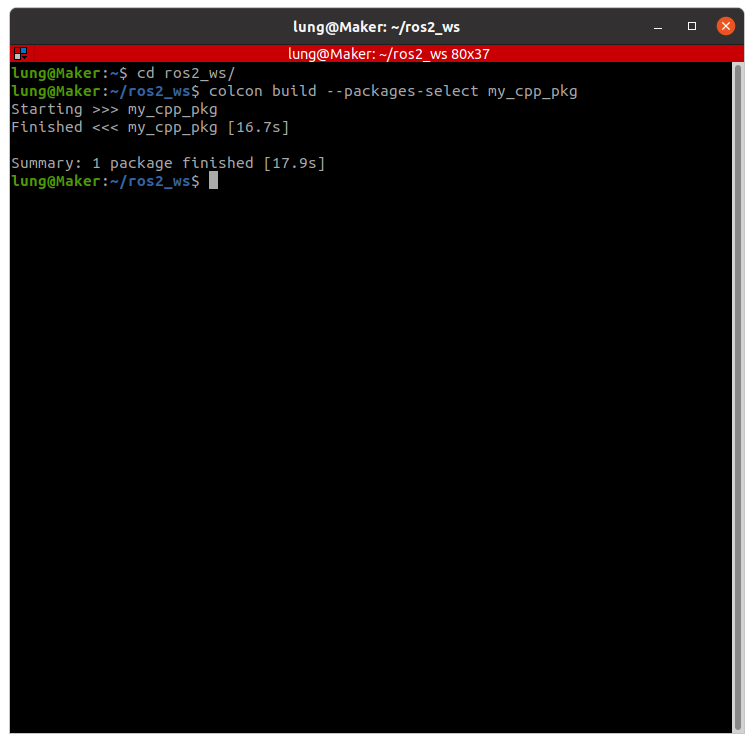
เข้าไปใน โฟลเดอร์ ros2\_ws

cd ros2\_ws/

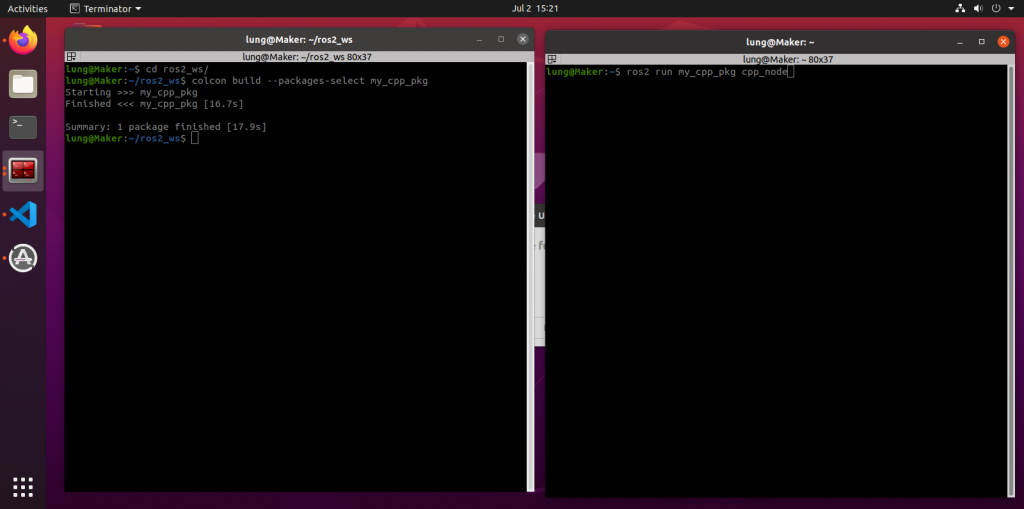


Build แพ็คเกจ my\_cpp\_pkg

colcon build --packages-select my\_cpp\_pkg

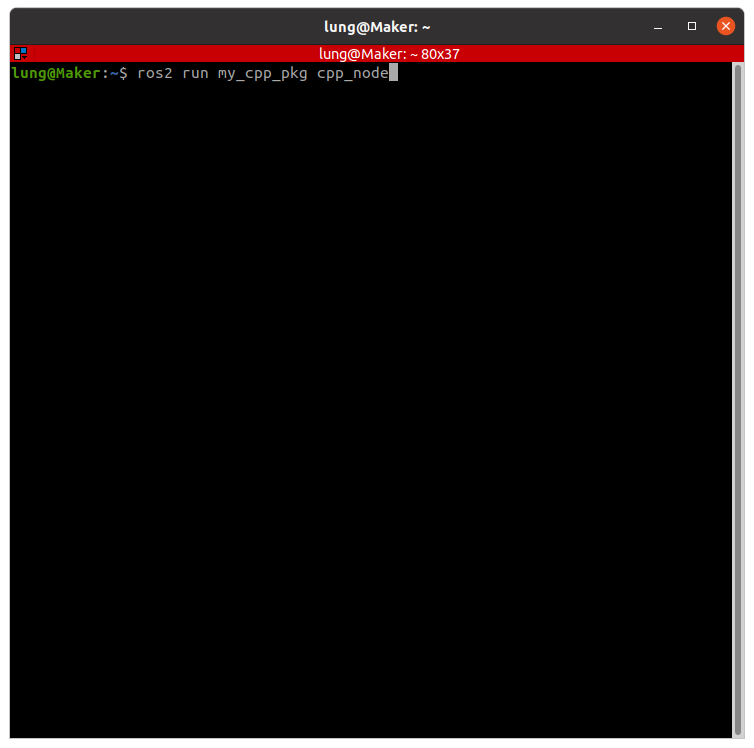


เปิด Terminator หน้าต่างที่ 2

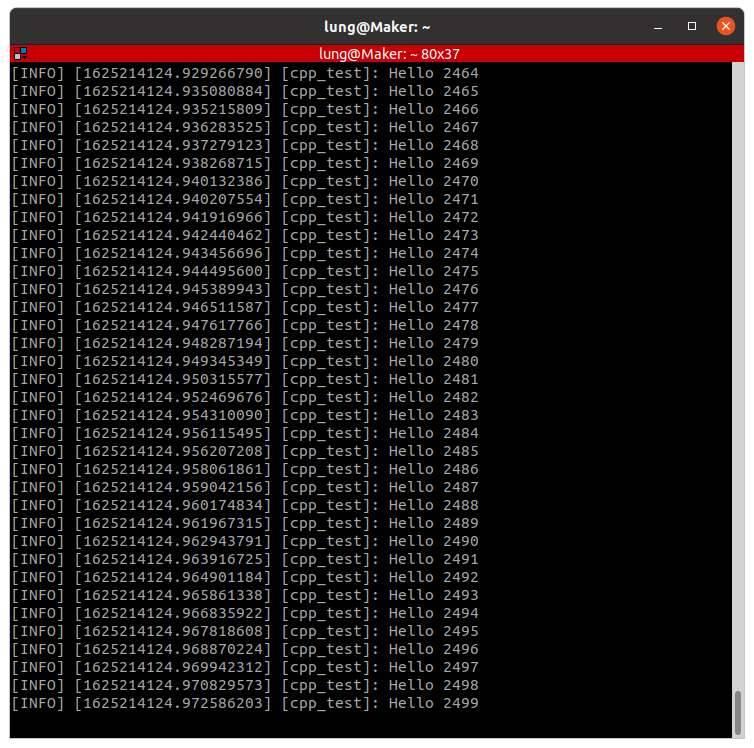


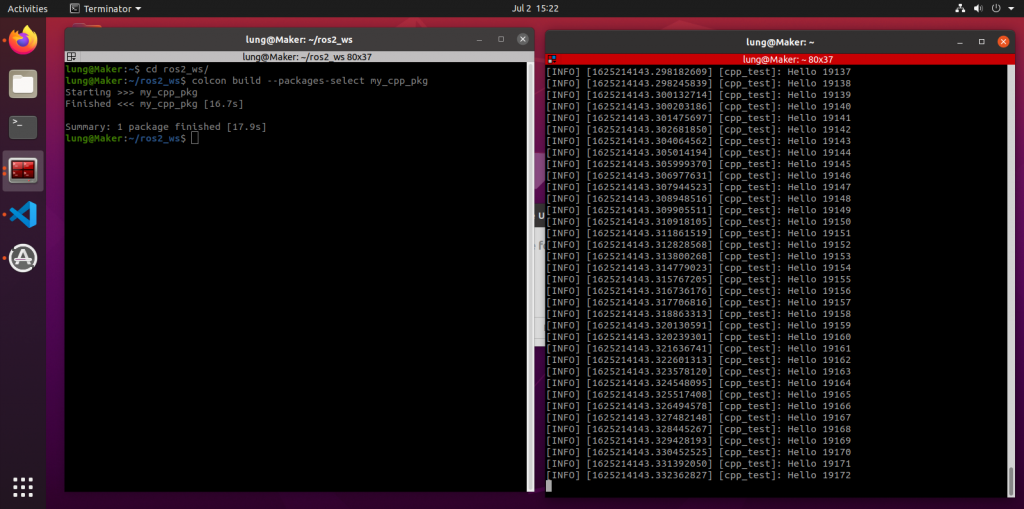
เรียกใช้งานด้วยคำสั่ง ros2 run ชื่อPackag ชื่อNode

ros2 run my\_cpp\_pkg cpp\_node



จะแสดงผลลัพธ์ การทำงาน





**3 : อธิบายโค้ด**  
ก่อนอื่นเรานำเข้าไลบรารี rclcpp จาก rclcpp เราจะสามารถดึงฟังก์ชันหลักของ ROS2 ได้มากมายเช่น nodes, topics, services, ฯลฯ

#include "rclcpp/rclcpp.hpp"

ในโปรแกรมหลักของเรา สิ่งแรกที่เราทำคือเริ่มต้นการสื่อสารระหว่าง ROS2 กับ rclcpp:: init (). คุณต้องผ่าน 2 อาร์กิวเมนต์ ซึ่งเป็นพารามิเตอร์ที่คุณได้รับจาก ฟังก์ชัน main() ส่วนนี้สำคัญมาก ต้องเรียก rclcpp:: init () ก่อนที่คุณจะสร้างโหนดใดๆ

int main(int argc, char \*\*argv)

{

rclcpp::init(argc, argv);

มาเขียนโค้ดขั้นต่ำที่เหมือนเดิมกันใหม่ แต่คราวนี้ด้วย OOP เราจะเพิ่มฟังก์ชันการทำงานบางอย่างหลังจากนั้น

int main(int argc, char \*\*argv)

{

rclcpp::init(argc, argv);

auto node = std::make\_shared<MyNode>();

RCLCPP\_INFO(node->get\_logger(), "Hello Cpp Node");

rclcpp::spin(node);

rclcpp::shutdown();

return 0;

}

เราประกาศวัตถุ ROS2 Timer เป็นแอตทริบิวต์ส่วนตัวของคลาส โปรดทราบว่า ROS2 นำเสนอประเภทที่มีประโยชน์บางอย่างแก่คุณ: คุณสามารถใช้ SharedPtr ซึ่งหมายความว่าในความเป็นจริงแล้ววัตถุจะอยู่ภายใน std::shared\_ptr.

rclcpp::TimerBase::SharedPtr timer\_;

เราเพิ่มการโทรกลับสำหรับตัวจับเวลานี้ เราทำให้วิธีนี้เป็นแบบส่วนตัว เนื่องจากจะเรียกจากภายในโหนดคลาสเท่านั้น

ภายในวิธีนี้เราเพียงแค่พิมพ์บางอย่างบนหน้าจอด้วย RCLCPP\_INFO (). เราใช้วิธีสืบทอดมา get\_logger () เพื่อรับตัวบันทึกของโหนดและการตั้งค่าทั้งหมดที่เข้ากันได้

private:

void timerCallback()

{

counter\_++;

RCLCPP\_INFO(this->get\_logger(), "Hello %d", counter\_);

}

ภายใน Constructor เราเริ่มต้นตัวจับเวลาด้วยวิธีอื่นที่สืบทอดมา: create\_wall\_timer (). เราจำเป็นต้องให้ 2 อาร์กิวเมนต์: ระยะเวลาระหว่างการเรียกกลับ 2 ครั้ง และฟังก์ชันที่จะเรียก ที่นี่เพื่อผ่านวิธีการเรียนที่เราต้องใช้ std::bind() การเรียกกลับจะเริ่มทำงานเมื่อโหนดเริ่มทำงาน

และอย่างที่คุณเห็น ไม่จำเป็นต้องเปลี่ยนแปลงอะไรในฟังก์ชันหลักของโปรแกรม ใช้คลาสที่สืบทอดมาจาก rclcpp::Node ทำให้แนวทางค่อนข้างเป็นแบบแยกส่วน

MyNode() : Node("cpp\_test"), counter\_(0)

{

RCLCPP\_INFO(this->get\_logger(), "Hello Cpp Node");

timer\_ = this->create\_wall\_timer(std::chrono::milliseconds(1),

std::bind(&MyNode::timerCallback, this));

}